

苏州大学实验报告

院、系	计算机学院	年级专业	软件工程	姓名	朱金涛	学号	2327406014
课程名称	机器学习综合实践					成绩	
指导教师	李俊涛	同组实验者	无		实验日期	2025年11月19日	

实验名称 基于Transformer架构的预训练模型完成机器翻译任务

一、实验目的

掌握 Hugging Face 生态系统：熟悉 transformers, datasets, evaluate 等核心库的使用方法。

理解 Seq2Seq 模型架构：通过 T5 (Text-to-Text Transfer Transformer) 模型完成从英文到中文的翻译任务。

掌握模型评估指标：理解并计算 BLEU 和 ROUGE 指标，用于定量评估机器翻译质量。

解决实际工程问题：学习处理模型加载错误、显存溢出 (OOM)、分词器兼容性及 Prompt 对模型输出的影响。

拓展多模态应用：体验 Transformer 架构在计算机视觉 (Vision Transformer) 领域的通用性。

二、实验内容

数据集： 使用 Helsinki-NLP/opus-100 数据集的 en-zh (英-中) 测试集，抽取前 2000 条数据。

核心模型 (NLP) : utrobinmv/t5_translate_en_ru_zh_small_1024 (基于 T5-small 微调的支持中/英/俄语的模型)。

对比模型 (失败对照组) : google-t5/t5-small (Google 原版模型)。

拓展模型 (CV) : google/vit-base-patch16-224 (用于图像分类)。

硬件环境： NVIDIA RTX 4060 Laptop GPU。

三、 实验步骤和结果

1. 环境配置与数据集加载

步骤描述： 在 Python 环境中安装 transformers, datasets, evaluate 等必要库。加载 Helsinki-NLP/opus-100 数据集的 en-zh (英-中) 测试集，并截取前 2000 条数据用于实验。

代码如下：

```
1 from datasets import load_dataset
2
3 # 加载数据集 (Opus-100 英译中测试集)
4 dataset = load_dataset("Helsinki-NLP/opus-100", "en-zh", split="test")
5 # 选取前 2000 条数据
6 test_data = dataset.select(range(2000))
7
8 # 提取输入与参考答案
9 english_inputs = [item['translation']['en'] for item in test_data]
10 chinese_references = [item['translation']['zh'] for item in test_data]
11
12 print(f"成功加载 {len(english_inputs)} 条数据。")
```

运行结果如下：

... 当前使用的设备： cuda
成功加载 2000 条英译中数据。

2. 模型加载与分词器兼容性修复

步骤描述： 一种利用了两种模型进行训练，一个是默认的 t5 模型，另一个是指定的

utrobinmv/t5_translate_en_ru_zh_small_1024 模型。 **难点解决：** 在 Windows 环境下直接加载会出现 SentencePiece 与 tiktoken 转换错误。通过在 AutoTokenizer 中添加 use_fast=False 参数，强制使用 Python 实现的慢速分词器，成功解决报错。

代码如下：

t5 模型：

```
1 # 4. 加载模型和分词器
2 #
3 model_name = "google-t5/t5-small"
4 tokenizer = AutoTokenizer.from_pretrained(model_name)
5 model = AutoModelForSeq2SeqLM.from_pretrained(model_name).to(device)
```

中文 t5：

```
1 from transformers import AutoTokenizer, AutoModelForSeq2SeqLM
2 import torch
3
4 device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
5 model_name = "utrobinmv/t5_translate_en_ru_zh_small_1024"
6
7 # 【关键修正】添加 use_fast=False 以解决 Windows 下的兼容性报错
8 tokenizer = AutoTokenizer.from_pretrained(model_name, use_fast=False)
9 model = AutoModelForSeq2SeqLM.from_pretrained(model_name).to(device)
10
11 print(f"模型已加载至: {device}")
```

运行结果如下：

开始使用 google-t5/t5-small 在 cuda 上对 2000 条数据进行推理...

翻译进度: 100% |██████████| 125/125 [02:44<00:00, 1.32s/it]

批量翻译推理完成！

3. prompt 修正与批量推理

步骤描述: 使用 GPU 对 2000 条数据进行批量推理。 **难点解决:** 初期使用通用提示词 translate English to Chinese: 导致模型输出俄语/德语。经调试，将提示词修正为模型特定的 **translate to zh:**，成功生成中文。同时，设置 num_beams=4 以保证翻译质量，设置 BATCH_SIZE=16 以适配显存。

代码如下：

```
● ● ●
1 from tqdm.auto import tqdm
2
3 # 【关键修正】使用模型特定的提示词，避免输出俄语
4 T5_PREFIX = "translate to zh: "
5 BATCH_SIZE = 16
6 predictions = []
7
8 # 批量推理循环
9 for i in tqdm(range(0, len(english_inputs), BATCH_SIZE), desc="翻译进度"):
10     batch_inputs = english_inputs[i:i + BATCH_SIZE]
11     # 拼接提示词
12     inputs_with_prefix = [T5_PREFIX + text for text in batch_inputs]
13
14     # Tokenize
15     tokenized_inputs = tokenizer(inputs_with_prefix, return_tensors="pt",
16                                  padding=True, truncation=True, max_length=128).to(device)
17
18     # 生成
19     outputs = model.generate(**tokenized_inputs, max_length=128, num_beams=4, early_stopping=True)
20     decoded_preds = tokenizer.batch_decode(outputs, skip_special_tokens=True)
21     predictions.extend(decoded_preds)
```

运行结果如下：

t5 模型的结果：

```
开始计算 BLEU 和 ROUGE 指标...
--- 翻译结果评估 (英译中 T5-small) ---
BLEU Score: 1.92
ROUGE Results (用于摘要和句子匹配的指标):
{
    "rouge1": 0.06681294171458776,
    "rouge2": 0.026142790232076964,
    "rougeL": 0.06121968727901171,
    "rougeLsum": 0.061102119549933526
}
```

中文 t5:

---- 正在计算评估指标 ----

 BLEU Score: 13.09

4. 翻译质量定量评估

步骤描述： 使用 sacrebleu 和 rouge 标准对模型生成的预测结果与参考答案进行比对。

核心代码：

```
● ● ●
1 import evaluate
2
3 # 格式化参考答案
4 formatted_references = [[ref] for ref in chinese_references]
5
6 # 计算 BLEU
7 bleu_metric = evaluate.load("sacrebleu")
8 bleu_results = bleu_metric.compute(predictions=predictions, references=formatted_references)
9
10 print(f"✅ BLEU Score: {bleu_results['score']:.2f}")
```

运行结果分析：

抽查其中部分翻译结果时，发现模型输出结果**并非中文**，而是该模型被训练时最常用的**德语**：

样本 1 (索引 0)：

【原文 (EN)】：The Global Programme of Action Coordination Office, with the f

【参考 (ZH)】：9. 《全球行动纲领》协调处得到比利时的财政支持，目前正帮助埃及、尼日利

【模型 (Pred)】：Das Büro für Koordinierung des globalen Aktionsprogramms unter

而中文 t5 模型的检测结果为中文且较为准确：

 [样本 #2 / 索引 18]

● 原文 (EN)：The movement of equipment by road from Port Sudan to Darfur, so

● 参考 (ZH)：从苏丹港到达尔富尔的路程大约1 400英里，目前装备平均需要7周时间才能运

● 预测 (T5)：目前，从苏丹港到达尔富尔大约1 400英里的公路运输设备平均需要7周。

5. “最有趣”的五个句子

(1) 原因：发现中文 t5 在很多情况下预测结果都能比参考答案要做到更优

 [样本 #2 / 索引 679]

● 原文 (EN)：And, um, she wants you to know that she is not at all scared.

● 参考 (ZH)：而且，嗯，她想让你知道 她是不是在所有的害怕。

● 预测 (T5)：而且，嗯，她希望你知道她根本不害怕。

(2) 原因：发现模型的预测结果甚至能更加贴近“**中文语境**”

 [样本 #3 / 索引 1978]

● 原文 (EN)：Strike weapons have not yet been deployed in outer space, and

● 参考 (ZH)：现在还没有在外层空间部署攻击武器，也没有作出部署这种武器的决定。

● 预测 (T5)：打击武器尚未部署到外层空间，也没有为此作出任何决定。

比如说这里的“尚未”，“为此”等词都不是对英文的直译，是带有中文常用的语气的。

(3) 哈哈，这个原因我就不说了，但是确实得给一个“最有趣”。(这数据集有点意思)

● 原文 (EN): - I like it when it's out of the ordinary personally.
● 参考 (ZH): - 別隱瞞 - 我感覺很好如果我們用不同尋常的做愛方法... ...就我個人而言
● 预测 (T5): 我喜欢它,因为它是普通人。

(4) 原因: 发现它反而不太能识别英文语境, "hello"就直接翻译为“你好”

[样本 #3 / 索引 1628]
● 原文 (EN): Hello, Thomas. You'd better come and hear this.
● 参考 (ZH): 嘿, 托马斯 你最好过来听听这个。
● 预测 (T5): 你好,托马斯,你最好来听这个。

(5) 原因: 该模型不能识别谚语, 但这确实也是常理之中。

[样本 #3 / 索引 1316]
原文 (EN): Bad things happen to people who kill their family with a hammer.
参考 (ZH): 我赔钱但不服气 战果很接近
预测 (T5): 坏事发生在用锤子杀死家人的人身上。

6. 在 Huggingface 上下载了一个 Vision Transformer 图像分类模型

步骤描述: 为了验证 Transformer 架构的通用性, 加载 google/vit-base-patch16-224 模型, 对一张猫的图片进行分类识别。

```
● ● ●  
1 from transformers import pipeline  
2 from PIL import Image  
3 import requests  
4  
5 # 加载 ViT 模型  
6 image_classifier = pipeline("image-classification", model="google/vit-base-patch16-224", device=0)  
7  
8 # 加载并识别图片  
9 url = "http://images.cocodataset.org/val2017/000000039769.jpg"  
10 image = Image.open(requests.get(url, stream=True).raw)  
11 results = image_classifier(image)  
12  
13 print(results)
```

实验结果:

模型成功识别出图片主体为 "**Egyptian cat**" (埃及猫), 置信度超过 90%。



正在识别图片内容...

✓ 识别结果 (Top 5):

- Egyptian cat: 93.74%
- tabby, tabby cat: 3.84%
- tiger cat: 1.44%
- lynx, catamount: 0.33%
- Siamese cat, Siamese: 0.07%

四、实验总结

本次实验基于 Hugging Face 生态系统，深入探索了 Transformer 架构在自然语言处理及计算机视觉领域的应用，重点完成了基于 Seq2Seq 架构的 T5 模型英中机器翻译任务。在实

验初期，我首先构建了对比实验，发现未经针对性微调的 Google 原版 T5-small 模型在处理中英翻译指令时存在严重偏差，错误地将英文翻译成了德语，这一“失败对照组”的现象直观地揭示了预训练模型在特定下游任务中进行微调的必要性。随后，我切换至 utrobinmv/t5_translate_en_ru_zh_small_1024 模型，并在此过程中解决了一系列实际工程问题。针对 Windows 环境下 SentencePiece 与 tiktoken 转换导致的加载错误，我通过在 AutoTokenizer 中添加 `use_fast=False` 参数，强制调用 Python 实现的慢速分词器，成功解决了分词器的兼容性难题。在推理阶段，我发现模型对提示词极其敏感，通用的翻译指令会导致模型输出俄语或德语，经过反复调试，我将 Prompt 修正为模型特定的“translate to zh:”，并配合 `BATCH_SIZE=16` 与 `num_beams=4` 的参数设置，在 NVIDIA RTX 4060 Laptop GPU 有限的显存资源下，成功完成了 2000 条数据的批量高质推理，有效避免了显存溢出问题。在对实验结果进行定量与定性评估时，通过 BLEU 和 ROUGE 指标的计算以及人工抽查，我发现微调后的中文 T5 模型表现优异。特别是在“最有趣”句子的分析中，通过对比模型预测结果与数据集参考答案，我发现模型生成的译文在很多情况下甚至比参考答案更贴近自然的“中文语境”，例如能够准确使用“尚未”、“为此”等具有中文语气的词汇，这表明模型不仅学会了直译，更掌握了一定的语言习惯；尽管模型在处理谚语或特定俚语（如将 hello 直译）时仍存在局限，但也侧面反映了已有的参考数据集本身可能存在的质量瓶颈。最后，为了验证 Transformer 架构的通用性，我成功加载了 Vision Transformer 模型完成了图像分类任务，准确识别出了埃及猫图片。本次实验不仅让我掌握了从数据加载、模型调试到性能评估的全流程技能，更深刻理解了 Transformer 架构在跨模态应用中的强大潜力与实际部署中的工程细节。